

---

# MAP Decoding with Parallelized Sliding Window Processing

---

U.S. Patent Application of:

Alan Gatherer,  
Tod D. Wolf,

Inventor  
Inventor

Texas Instruments Incorporated,

Assignee

Attorney's Docket No. TI-30024

## MAP Decoding with Parallelized Sliding Window Processing

### Background and Summary of the Invention

The present application relates to wireless communication, and more particularly to turbo decoding and the like.

#### Background: Error Correction

Coded digital communication systems use error control codes to improve data reliability at a given signal-to-noise ratio (SNR). For example, an extremely simple form (used in data storage applications) is to generate and transmit a parity bit with every eight bits of data; by checking parity on each block of nine bits, single-bit errors can be detected. (By adding three error-correction bits to each block, single-bit errors can be detected and corrected.) In general, error control coding includes a large variety of techniques for generating extra bits to accompany a data stream, allowing errors in the data stream to be detected and possibly corrected.

#### Background: Trellis Coding

One of the important techniques for error control is trellis coding. In this class of techniques some constraints are imposed on the sequence of symbols, so that certain symbols cannot be directly followed by others. The constraints are often defined by a geometrical pattern (or "trellis") of allowed and disallowed transitions. The

existence of constraints on the sequence of symbols provides some structure to the data sequence: by analyzing whether the constraints have been violated, multiple errors can be corrected. This is a very powerful class of coding techniques; the constraint geometry can be higher dimensional, or algebraic formulations can be used to express the constraints, and many variations can be used.

### **Background: Turbo Coding**

The encoder side of a turbo coding architecture typically uses two encoders, one operating on the raw data stream and one on a shuffled copy of the base data stream, to generate two parity bits for each bit of the raw data stream. The encoder output thus contains three times as many bits as the incoming data stream. This "parallel concatenated encoder" (or "PCE") configuration is described in detail below.

The most surprising part of turbo coding was its decoding architecture. The decoder side invokes a process which (if the channel were noiseless) would merely reverse the transformation performed on the encoder side, to reproduce the original data. However, the decoder side is configured to operate on soft estimates of the information bits and refines the estimates through an iterative reestimation process. The decoder does not have to reach a decision on its first pass, but is generally allowed to iteratively improve the estimates of the information bits until convergence is achieved.

## Background: MAP Decoders

MAP decoding is a computationally intensive technique, which has turned out to be very important for turbo decoding and for trellis-coded modulation. "MAP" stands for "maximum a posteriori": a MAP decoder outputs the most likely estimate for each symbol in view of earlier AND LATER received symbols. This is particularly important where trellis coding is used, since the estimate for each symbol is related to the estimates for following symbols.

By contrast, a maximum-likelihood ("ML") decoder tries to compute the transmitted sequence for which the actually received sequence was most likely. These verbal statements may sound similar, but the difference between MAP and ML decoding is very significant. ML decoding is computationally simpler, but in many applications MAP decoding is required.

MAP decoding normally combines forward- and back-propagated estimates: a sequence of received symbols is stored, and then processed in one direction (e.g. forward in time) to produce a sequence of forward transition probabilities, and then processed in the opposite direction (backward in time) to produce a sequence of backward transition probabilities. The net estimate for each symbol is generated by combining the forward and backward transition probabilities with the data for the signal actually received. (Further details of this procedure can be found in OPTIMAL DECODING OF LINEAR CODES FOR MINIMIZING SYMBOL ERROR RATE, Bahl, Cocke, Jelinek, and Raviv, *IEEE Transactions on Information Theory*, 1974, which is

hereby incorporated by reference.)

The combination of forward and backward computation requires a substantial amount of memory. Since the blocks in advanced cellular communications can be large (e.g. 5120 symbols), the memory required to store a value for each possible transition for each symbol in a block is large. To reduce the memory requirements during decoding, each block of data may be divided into many smaller blocks (e.g. 40 blocks of 128 symbols) for MAP decoding.

The trellis encoding is done on a complete block of data, so that starting and ending states are known for the complete block. However, the starting and ending states are not known for the intermediate blocks.

This presents a problem for accurate process of these smaller blocks, but it has been found that simply iterating the forward estimation process for a few symbols before the start of each block will ensure that processing of the first symbol in the block starts from a good set of initial values.

### **MAP Decoding with Pipelined Windowed Processing**

The present application discloses a technique for sub-block processing, in a MAP decoding, which uses pipelining. Processing of alphas is begun, in parallel with processing of betas. Preferably each stage of processing is further internally parallelized; but the pipelining of forward-propagated processing with back-propagated processing provides an additional degree of net improvement in throughput.

Advantages of the disclosed methods and structures, in various

embodiments, can include one or more of the following:

- Faster processing;
- less memory;
- more iterations possible in a turbo decoder.

## Brief Description of the Drawings

The disclosed inventions will be described with reference to the accompanying drawings, which show important sample embodiments of the invention and which are incorporated in the specification hereof by reference, wherein:

**Figure 1** shows a block diagram of a turbo decoder.

**Figure 2** shows a block diagram of a MAP decoder that uses parallel sliding window processing.

**Figure 3** shows a block diagram of the beta generation block within the MAP decoder.

**Figure 4** shows a block diagram of the alpha generation block within the MAP decoder.

**Figure 5** shows a block diagram of the extrinsic generation block within the MAP decoder.

**Figure 6** is a timing chart of the pipelining within the beta block.

**Figure 7** shows the timing offset between generation of alpha and beta sliding window blocks.

**Figure 8** shows the correspondence between the alpha and beta sliding window blocks, with prologs.

**Figure 9** shows an example of the order in which beta and alpha bits are processed.

## Detailed Description of the Preferred Embodiments

The numerous innovative teachings of the present application will be described with particular reference to the presently preferred embodiment. However, it should be understood that this class of embodiments provides only a few examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily delimit any of the various claimed inventions. Moreover, some statements may apply to some inventive features but not to others.

50  
49  
48  
47  
46  
45  
44  
43  
42  
41  
40  
39  
38  
37  
36  
35  
34  
33  
32  
31  
30  
29  
28  
27  
26  
25  
24

Concurrent operation of system hardware allows simultaneous processing of more than one basic operation. Concurrent processing is often implemented with two well known techniques: parallelism and pipelining.

Parallelism includes replicating a hardware structure in a system. Performance is improved by having multiple structures execute simultaneously on different parts of a problem to be solved.

Pipelining splits the function to be performed into smaller pieces and allocates separate hardware to each piece. More information on parallelism and pipelining can be found in the "The Architecture of Pipelined Computers," by Kogge, which is hereby incorporated by reference.

**Figure 1** shows a block diagram of a turbo decoder. Two main blocks, the turbo controller **102** and the MAP decoder **104**, are shown.

The turbo controller 102 stores the data streams (X, the systematic data 106; P, the parity data 108; and A, the A PRIORI data 110) that serve as input for the MAP decoder 104 and controls the order in which the data is input in the MAP decoder 104. The diagram shows the three data streams being input twice each in the MAP decoder 104. Two separate sets of input data are required because the alpha and beta generation blocks require the data inputs in reverse order. The extrinsic output of the MAP decoder 104 is returned to the controller 102 for another decoding iteration.

**Figure 2** shows a block diagram of a MAP decoder 104 that uses parallel sliding window processing. A MAP decoder 104 receives the scaled systematic data signal 106, the scaled parity data signal 108, and the A PRIORI signal 110 as its input. There are N number of X signals 106, where N is the size of the interleaver. N X signals 106 are applied for each beta 208 and alpha 210 state vector, which are the respective outputs of the beta 202 and alpha 206 blocks. During beta generation, X 106 is applied in reverse order, and during alpha generation, X 106 is applied in forward order. There are also N number of P signals 108. N P signals 108 are applied for each alpha 210 and beta 208 vector. During beta generation, P 108 is applied in reverse order, and during alpha generation it is applied in forward order. The A PRIORI 110 is either the interleaved or deinterleaved extrinsic data from the previous MAP decoder operation. There are N A PRIORI signals 110, and one A PRIORI 110 is applied for each beta 208 and alpha 210 vector. A PRIORI 110 is applied the same directions as X 106 and P 108 for beta

and alpha generation.

The beta generation section 202, shown in more detail in **Figure 3**, receives inputs X 106, P 108, and A 110. It generates the beta state vector 208, which is stored in beta RAM 204. The alpha generation section 206 receives inputs X 106, P 108, and A 110 (but in reverse order relative to the beta input). The alpha generation block 206, shown in greater detail in **Figure 4**, generates the alpha state vector 210. The outputs 208, 210 of the alpha and beta generation sections serve as inputs for the extrinsic generation section 212, shown in **Figure 5**. These data streams must be properly sequenced with the parity stream P 108 before being input to the extrinsic section 212.

**Figure 3** shows the beta generation stages. First, during the MAP reset state, the registers are set to their initial conditions for the beta state vector 208. The beta signal 208, X 106, P 108, and A 110 are summed together by the adder tree 302 according to the trellis used to encode the data. (In the preferred embodiment, an 8 state trellis is used). The results are stored in registers 310. In the second stage, the results of the adder 302 are applied to the 8 MAX\* blocks 304, and are then stored in the MAX\* registers 312. Next, the unnormalized outputs advance to two separate normalization stages 306, 308, each of which has a register 314, 316 to store its results. Thus the total process has 4 stages within the feedback loop of the beta generation block 202, which require 4 clock cycles to complete. This latency (the 4 clock cycles) determines the level of pipelining available.

The alpha generation section 206 is shown in **Figure 4**. First, the

registers are set to their initial conditions. Then the data inputs are summed together by the adder 402, and the results are stored in registers 412. These are then input to the MAX\* blocks 406 and stored in MAX\* registers 414. The alpha generation section 206 also has two normalization stages 408, 410, each with their own registers 416, 418. The latency of the alpha generation stage 206 is thus 4, allowing 4 levels of pipelining to be implemented.

Operating in parallel with the alpha generation section is the extrinsic generation section 212, shown in **Figure 5**. Alpha 210, beta 208, and P 108 are summed together by the adders 502 according to the trellis used, and the results are stored in registers 510. In the second stage, these results are applied to the MAX\* blocks 504, and then stored in the MAX\* registers 512. These results are again applied to MAX\* blocks 504 and then stored in registers 508. The result is summed and stored in another register 514, and the output is the extrinsic signal 214.

### **Parallelism with Sliding Windows**

A sliding window approach basically consists of dividing the N sized block of incoming data into several smaller blocks. Each of these smaller blocks is called a sliding window block. These sliding window blocks are MAP decoded each independently, with a prolog for both the alpha and beta vectors. The decoding for the individual alpha and beta sliding window blocks is done in parallel. Since the initial conditions are not known for the individual sliding window blocks, the

prologs are used to reach a good set of initial values.

By starting the update of the alpha at a point sufficiently inside the previous block and starting the update of the beta at a point sufficiently inside the next block, the decoder can "forget" the initial conditions and converge before it begins operating on the actual data. the prolog section size used is generally 3 or 4 times the number of states in the trellis. The first alpha and last beta sliding block will originate from a known state, and the size of their respective prolog sections will be 3 for an 8 state trellis (for example).

The innovative alpha prolog allows parallel processing of both the alpha and beta sliding window blocks of data. Depending on the specific implementation used, each update of alpha or beta takes a few clock cycles to run (4 clock cycles in the above embodiment). This latency determines the degree of pipelining possible in the system. In the preferred embodiment, there are four levels of pipelining within each alpha and beta block (meaning the data within each of the alpha and beta generation stages is pipelined, or broken into separate sets of data and independently operated on by successive stages within the beta generation section). There is also a degree of parallelism between the alpha and beta blocks themselves, meaning these two sections operate simultaneously to produce extrinsic input.

The alpha and beta vector generation processes are divided into multiple stages, as shown above. These stages are within the iteration loops of the alpha and beta vector generation, shown in **Figures 3** and **4**. The number of stages would be equal to the latency for a particular

architecture. In the preferred embodiment, these stages are the Adder, the MAX\*, and two Normalization stages. The latency of these stages dictates the degree of parallel processing possible. For example, in the preferred embodiment this latency is 4, meaning 4 sliding-window blocks can be processed in parallel. Thus, 4 sliding-window blocks make up one sub-block.

The pipelining of the sliding blocks is shown in **Figure 6**. During the first clock cycle, beta0 (the first sliding block) enters the adder stage. In the second clock cycle, beta0 enters the MAX\* stage, and beta1 enters the adder stage. In the third clock cycle, beta0 enters the first normalization stage (the third stage of beta generation), beta1 enters the MAX\* stage, and beta2 enters the adder stage. Next, beta0 enters the second normalization stage, beta1 enters the first normalization stage, beta2 enters the MAX\* stage, and beta3 enters the adder stage. The intermediate values for each stage are stored in registers, as shown above.

Either the beta or alpha stages are stored in memory so that the data input to the extrinsic section can be synchronized. In the preferred embodiment, beta processing begins one sub-block before alpha processing (Note that this staggering could be eliminated by adding another RAM block to store the alpha outputs.) This staggering is shown in **Figure 7**. The first sub-block (which is a number of sliding blocks equal to the latency of the architecture--4 in the preferred embodiment) of the beta section can be processed while the alpha section is idle. Next, the second set of sliding blocks of the beta section

(i.e., the second sub-block) is processed while the first set of sliding blocks of the alpha section are processed. The extrinsic sections are processed in parallel with the alpha section. This reduces the memory requirement for storing both the alpha and beta state vectors because the alpha outputs can be directly applied to the extrinsic as they are generated. Since the extrinsic generates output (and requires input) one sub-block at a time, the beta RAM only needs to store one sub-block of data at a time. (Note that the alpha and beta processing could be reversed. This would require the alpha outputs to be stored in memory, and beta and the extrinsic blocks would run in parallel.)

**Figure 8** shows the correspondence between alpha and beta sliding-window blocks. The entire data block consists of  $N$  symbols plus a number of tail bits. This block is broken into sub-blocks, which are further divided into sliding-window blocks. One sliding-window block is processed per clock cycle. Each sliding-window block includes a prolog. The beta prologs consist of several symbols to the right of the sliding window. The alpha prolog consists of the several bits to the left of the sliding window. This is shown by the overlap between successive sliding blocks in the figure. Each beta sliding window is processed in reverse relative to the alpha sliding blocks.

**Figure 9** shows an example of the order in which beta and alpha bits are processed. This example assumes a sliding window size of 100, a prolog length of 24, and 4 sliding windows per sub-block. The sliding block beta0 begins at the start of the prolog at bit 123. Next, the prolog ends at bit 100. The reliability data begins at bit 99, and ends at bit

zero. The alpha sliding blocks are similarly divided. (Note the first two entries for alpha do not exist, because there is no prolog for the beginning of the block since the start and end points are known.)

The extrinsic cannot be processed in parallel with both the alpha and beta generation processes, because the extrinsic input data, which requires data from alpha, beta, and the parity data, must be input in a certain order. The following shows the indexing of the extrinsic input. E0 (corresponding to alpha0 and beta0) goes from bit 0 to 99. E1 goes from 100 to 199, and so on, given a sliding window size of 100. The input required by this example would be as follows. In the first clock cycle, the soft estimate data relating to bit 0 from alpha, beta, and P are input to the extrinsic. In the second clock cycle, data associated with bit 100 from the three inputs is required. In the third clock cycle, the data associated with bit 200 is required. In the fourth clock cycle, the data associated with bit 300 is required. In the fifth clock cycle, the input reverts back to the data associated with bit 1 (the first clock cycle input shifted one bit). In the next cycle, the bit 101 data, and so on. Thus the betas must be stored in RAM after they are generated, because they are generated in a different order than the alpha bits and parity bits, and are not required at generation as are the alpha bits and parity bits. When the corresponding alphas and betas have been generated, the extrinsic may be calculated.

### **Definitions:**

Following are short definitions of the usual meanings of some of

the technical terms which are used in the present application. (However, those of ordinary skill will recognize whether the context requires a different meaning.) Additional definitions can be found in the standard technical dictionaries and journals.

**MAX\***: MAX\* is a maximum finding approximation for the natural log function, given by the following equation:

$$\ln[e^A + e^B] \approx \text{MAX}^* = \text{MAX}(A + B) + f(|A - B|)$$

where  $f(A-B)$  is a correction term. A lookup table is usually used for this value, which makes the above expression an approximation. If the expression

$$\ln[1 + e^{-|A-B|}]$$

is used instead of a lookup table, then the MAX\* definition becomes an exact equality, not an approximation.

**MAP decoder**: Maximum A-Posteriori. MAP decoders use a detection criterion that leads to the selection of  $x$  that maximizes the probability  $p(x|r)$  of a symbol  $x$  given the received information  $r$ .

**Extrinsic**: Outputs of decoders that estimate the value of a decoded bit.

Extrinsics are usually soft estimates.

## **Modifications and Variations**

As will be recognized by those skilled in the art, the innovative concepts described in the present application can be modified and varied over a tremendous range of applications, and accordingly the scope of patented subject matter is not limited by any of the specific exemplary teachings given, but is only defined by the issued claims.

0  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
598  
599  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
689  
690  
691  
692  
693  
694  
695  
696  
697  
697  
698  
699  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
797  
797  
798  
799  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
889  
890  
891  
892  
893  
894  
895  
896  
897  
897  
898  
899  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
988  
989  
989  
990  
991  
992  
993  
994  
995  
995  
996  
997  
997  
998  
999  
999  
1000  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1095  
1096  
1097  
1097  
1098  
1099  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1188  
1189  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1195  
1196  
1197  
1197  
1198  
1199  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1288  
1289  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1295  
1296  
1297  
1297  
1298  
1299  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1388  
1389  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1395  
1396  
1397  
1397  
1398  
1399  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1488  
1489  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1495  
1496  
1497  
1497  
1498  
1499  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1588  
1589  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1595  
1596  
1597  
1597  
1598  
1599  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1688  
1689  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1695  
1696  
1697  
1697  
1698  
1699  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1788  
1789  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1795  
1796  
1797  
1797  
1798  
1799  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1888  
1889  
1889  
1890  
1891  
1

of other variables. The disclosed innovations are intended to cover all such variations in implementation.

The disclosed innovations of the present application are applicable to any MAP architecture. For instance, any implementation of the disclosed inventive concepts in turbo decoders which use MAP decoders is within the contemplation of the invention. Any MAP operations, e.g., MAP equalization, are within the contemplation of the present application. MAP equalization is the process of describing the channel function as the data input to the channel constrained on a trellis to produce the observed output. The input to the channel can then be estimated in a maximum a priori sense by applying a MAP decode to the trellis diagram and the observed channel output. This is useful if (a) soft output is required from the equalizer, (b) a more accurate estimate of the input to the channel is required than can be got using a linear filter or equalizer, or (c) an iterative joint decode of the channel and the applied FEC is required. In general, MAP finds use in any situation where the data observed is known to have been generated by input to a linear trellis.

Likewise, MAP architectures with software, as well as hardware, implementations is within the contemplation of the invention. In today's DSPs very high processing rates are achieved by using deep pipelining of the data path. This means the DSP cannot be efficiently used in a feedback process such as beta and alpha updates. Using the present invention allows several blocks to be simultaneously processed by the DSP in a pipelined fashion, which considerably

speeds up the operation in a deeply pipelined DSP architecture.

Further background material on the state of the art in MAP decoders and coding can be found in TURBO CODING, by Heegard and Wicker; TRELLIS CODING, by Schlegel; ERROR CONTROL SYSTEMS, by Wicker, and AN INTUITIVE JUSTIFICATION AND A SIMPLIFIED IMPLEMENTATION OF THE MAP DECODER FOR CONVOLUTIONAL CODES, Andrew Viterbi, IEEE Journal on Selected Areas of Communications, Vol. 16, No. 2, February 1998, all of which are hereby incorporated by reference.

TOP SECRET//COMINT